//

// main.cpp

// Lab3

// Stephanie Marin Velasquez

// Professor: Yumei Huo

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

//

//// REVERSE A NUMBER (INTEGER BY ITERATIVE PROGRAM)

#include<iostream>

using namespace std;

void reverse (int num)

{

int rev;

while(num != 0)

{rev=(num%10);

cout<< rev;

num=num/10;}

}

///// REVERSE A NUMBER ( INTEGER BY RECURSIVE PROGRAM)

void rev(int num)

{

int r;

if(num!=0)

{

r=num%10;

cout<<r;

return rev(num/10);

}

}

int main()

{

int num;

cout<<"Enter an integer: "<< endl;

cin>> num;

cout<< " \n Integer reversely is: ";

//reverse(num);

cout<< endl;

rev(num);

cout<< endl;

}

// Lab3

// Stephanie Marin Velasquez

// Professor: Yumei Huo

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

//// ADT array-Based polynomial TRY 6

#include<iostream>

#include<cmath>

using namespace std;

class polynomial {

public:

polynomial(int degree=0);

~polynomial(){}

void setCoef()const;

int getD()const;

int getX()const;

polynomial( const polynomial&);

void makePoly ();

void multyPoly(int x);

void addPolys();

void copy(const polynomial& poly);

friend ostream& operator<< ( ostream& out, const polynomial& x);///

friend istream& operator>> (istream& , polynomial& poly); ///

//polynomial operator = (const polynomial& poly);

friend polynomial operator \*( polynomial&, polynomial&);

//polynomial& operator= ( const polynomial& rhs);

//const polynomial operator+(const polynomial& rhs);

private:

int d;

int\* coeff;

}; polynomial poly, poly2;

int polynomial::getD()const{return d;}

polynomial::polynomial( int degree)

{

d = degree;

coeff = new int [d+1];

}

polynomial::polynomial(const polynomial& poly)

{

if (poly.d>poly2.d)

d=poly.d;

else d=poly2.d;

if(poly.d||poly2.d)

{for(int i=0;i<poly.d+1;i++)

{ coeff[i]=poly.coeff[i];

// cout<< poly.coeff[i] << "\*"<<poly.x;

//multi= poly.coeff[i]\*(poly.x\*poly.x);

}}

}

istream& operator >> (istream& in, polynomial& poly)

{

int deg;

in>>deg;

poly=deg;

return in;

}

ostream& operator<<(ostream& out, polynomial& poly)

{

int let;

out<< poly;

poly=let;

return out;

}

void polynomial::setCoef()const

{

for (int i=0; i< d+1; i++)

{ cout<< "Enter the coeficient for term "<< i+1 <<endl;

cin>> coeff[i];

cout<< "You entered "<< coeff[i]<<endl;

}

}

void polynomial:: makePoly()

{

for ( int p=d; p>-1; p--)

{

if ( coeff [ d-1 ] !=0 )

{

if( (p>=2 || p==d) && ( p !=0 & p!=1) && (coeff[d-p]>1))

cout<< coeff [d-p]<< "x^" << p <<"+ ";

if( (p>=2 || p==d) && ( p !=0 & p!=1)&& (coeff[d-p]==1))

cout<< "x^" << p <<"+ ";

if (p == 1 && coeff[d-p]!= 1)

cout<< coeff[ d-p]<< "x+ ";

if (p == 1 && coeff[d-p]== 1)

cout<< "x+ ";

if (p == 0)

cout<< coeff [d-p];

if ( coeff[d-p] == 0)

continue;

}

}

}

void polynomial:: multyPoly(int x) /// polynomial with value of x

{

int sum=0;

for ( int p=d; p>-1; p--)

{

sum+= coeff[d-p]\* pow(x,p);

} cout<< endl;

cout<< sum<<endl;

}

void polynomial::addPolys()

{

if (poly.d>poly2.d)

d=poly.d;

else d=poly2.d;

int \*ans = new int[d];

//addition

for(int i=0; i<d+1; i++)

{

ans[i]= poly.coeff[i]+poly2.coeff[i];

}

for (int i=d+1; i<poly2.d;i++)

{

ans[i]=poly.coeff[i];

}

for (int p=d; p>=0; p--)

{

// int sumcoef;

if ( poly.coeff [ d-1 ] !=0 && poly2.coeff[d-1] )

{

if( (p>=2 || p==d) && ( p !=0 & p!=1))

cout<< ans[p]<< "x^" << p <<"+ ";

if (p == 1)

cout<< ans[p]<<"x+ ";

if (p == 0)

cout<< ans[p];

//if ( ans[p] == 0)

// continue;

}

}

}

void printA()

{

cout<< "Enter the polynomial degree \n"<< ": ";

cin>>poly;

cout<< " The number you entered is \n "<< ": ";

cout<< poly.getD();

cout<<endl;

poly.setCoef();

poly.makePoly();

}

void printB()

{

int x;

cout<< " \n Enter a value for x\n"<< ":";

cin>>x;

cout<< "The number you entered is \n"<<":";

cout<<x;

cout<<endl;

poly.multyPoly(x);

}

void printC()

{

cout<< "Enter the polynomial degree \n"<< ": ";

cin>>poly2;

cout<< " The number you entered is \n "<< ": ";

cout<< poly2.getD();

cout<<endl;

poly2.setCoef();

poly2.makePoly();

cout<<endl;

poly2.addPolys();

}

int main()

{

cout<< ".... Part A....\n"<<endl;

printA();

cout<< "....Part B....\n"<<endl;

printB();

cout<< "....Part C....\n"<<endl;

printC();

cout<< endl;

}